Lecture 7 Scribe

Camera Calibration 2005/04/05

R93922043 高海峰
R93922030 廖守鈞
P93922002 林宗勳

在做 Camera Calibration 和 Bundle Adjustment 之前會用到 Nonlinear Least square 的技巧，因此在這裡會先介紹一下 Nonlinear Least square 的技巧。但是在介紹 nonlinear least square 之前，我們先來複習一下 least square 的技巧。

<u>Least square methods:</u>

Least Squares Problem

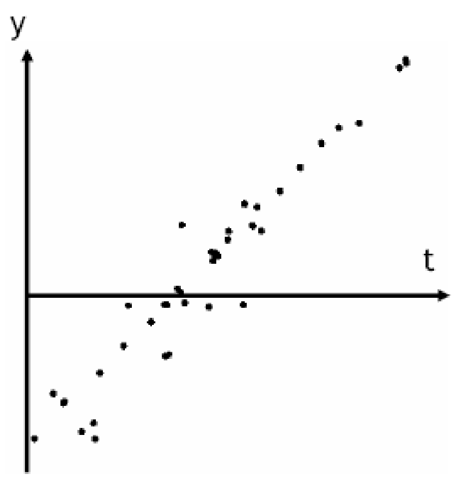
Find $\mathbf{x}^*$, a local minimizer for

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 ,$$

where $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \ldots, m$ are given functions, and $m \geq n$.

Least Square Problem 的目的就是要找到一個 X*，使得 F(X)的值會最小。F(X)是寫成一些 quadratic form 相加的形式。在給定一個 Model 的情形下，和一大堆 Data 的情形下，我們希望找到一個最佳的參數，使得 F(X)的值爲最小。

- **Example: Linear least square problem**



在這個例子中，給定一些點，我們想要知道這些點是由哪一個 Model 所生成的。在這邊我們假設這些資料是由一個線性的 Model 所產生的。
我們假設這個 Model 的形式如下：

$$y(t) = M(x,t) = x_0 + x_1 t$$

給定 Data (t1,y1), (t2,y2), …, (tn,yn)，我們要找一個最佳的參數 X* = (x_0*, x1*)，使得我們用這個最佳的參數所預測出來值和給定的 Data 之間的均方誤差最小。

$$f_i(x) = y_i - M(x,t_i) = y_i - x_0 - x_1 t_i$$

fi(x)是 residue function,它代表的就是給定的 Data 和我們用 model 預測出來的值的差異。

我們要 Minimize 下面這個函數：

$$\begin{aligned} F(X) &= \frac{1}{2}\sum_{i=1}^{m}(f_i(x))^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}(y_i - x_0 - x_1 t_i)^2 \end{aligned}$$

對 x_0, x_1 做偏微分，我們可以得到

$$\frac{\partial F(X)}{\partial x_0} = \sum_{i=1}^{m}(y_i - x_0 - x_1 t_i) \quad = 0$$

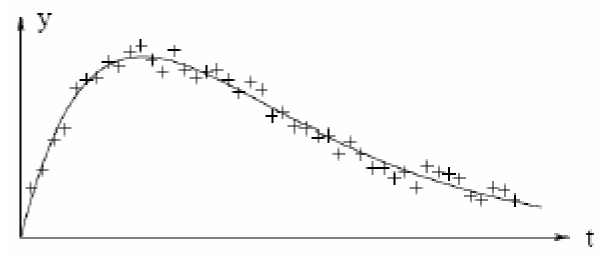
$$\frac{\partial F(X)}{\partial x_1} = -\sum_{i=1}^{m}(y_i - x_0 - x_1 t_i)t_i \quad = 0$$

因為 t_i , y_i 為已知，所以上面的式子為一個二元一次聯立方程式，用一般解線性方程式的方法求解即可。

值得注意的一點就是在這裡的 linear 指的是 Model 和 Model 的參數是線性關係。例如：$M(x,t) = x_0 + x_1 t + x_2 t^3$ 也是是 linear 。雖然這個函數和 t 不是一個 linear 的關係，但是若固定 t 的值，只考慮 x 的話，它是一個 linear function，因此我們還是可以用 linear least square 的方法。

- **Nonlinear Least Square**

Nonlinear least square 的問題就沒有這麼單純了，因為 M(x,t)和 x 的關係不是線性。考慮以下的例子：



$$model\ M(\mathbf{x}, t) = x_3 e^{x_1 t} + x_4 e^{x_2 t}$$

$$parameters\ \mathbf{x} = [x_1, x_2, x_3, x_4]^{\top}$$

$$residuals\ \ f_i(\mathbf{x}) = y_i - M(\mathbf{x}, t_i)$$
$$= y_i - x_3 e^{x_1 t_i} - x_4 e^{x_2 t_i}$$

在這個 model 中，M(x,t)和它的參數 x1,x2,x3,x4 的關係都不是線性的，所以如果我們把 residual function fi(x)對 x 去做偏微分，我們會得到一大堆非線性方程式的聯立方程式。這樣的方程式並不是很容易求解。

在 least square 的問題中，我們要 minimize F(x)，但因爲 F(x)是一個非線性方程式，這樣的方程式要求得最佳解很困難，所以我們通常只設法去找到局部極小值(Local Minimum)。

我們現在先試圖在給定 x 的情形下，從 x 的周圍找到一個點 x*，使得 F(x*)的值比 F(x)來的小。

Local Minimizer Given $F: \mathbb{R}^n \mapsto \mathbb{R}$. Find $\mathbf{x}^*$ so that $F(\mathbf{x}^*) \le F(\mathbf{x}) \quad \text{for} \quad \|\mathbf{x} - \mathbf{x}^*\| < \delta$.

爲了解決這樣子的問題，我們假設 F 是二次可微，而且是 smooth 的。因此我們可以得到 F 的泰勒展開式。我們假設 h 很小，所以忽略三次項以上的值。

$$F(x+h) = F(x) + h^T g + \frac{1}{2} h^T A h + O(\|h\|^3)) \quad \text{-----(1)}$$

其中 g 是 F 的 gradient：

$$g \equiv F'(x) = \begin{bmatrix} \frac{\partial F(x)}{\partial x_1} \\ \vdots \\ \frac{\partial F(x)}{\partial x_n} \end{bmatrix}$$

H 則是 Hessian Matrix

$$H \equiv F''(x) = \left[\frac{\partial^2 F(x)}{\partial x_i \partial x_j} \right]$$

假設這個函數在一個很小的區域內是 smooth 的，則我們可以用上面的泰勒展開式來近似這個函數。其中 g 和 H 是 F(x)在 x 這一點的一階和二階導數，在固定 x 的情形下，g 和 H 爲常值。因此(1)可以看成是 h 的二次式(quadratic form)：

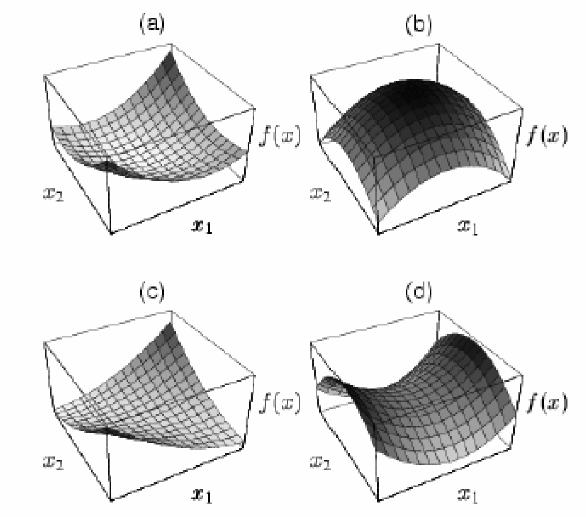
$$F(x+h) = f(h) = c - b^T h + \frac{1}{2} x^T A x \quad \text{-----(2)}$$

爲了得到(2)的局部極小值，我們把(2)對 h 做偏微分

$$\frac{\partial f(h)}{\partial h} = A^T h - b = 0$$

上式爲一個線性系統，可以解得 $h^* = (A^T)^{-1} b$

f(h)的形狀和 A 有很大的關係。當 A 爲 positive definite(positive definite)，f(h)的函數圖如下面圖(a)所示。當 A 爲負定(negative definite)，f(h)的函數圖如下面圖(b)所示。當 A 爲 singular 時，f(h)的函數圖如圖(c)所示。



我們希望 f(h)有極小值，所以理想的圖形應該是圖(a)。因此 A 得要是 positive definite 矩陣。

- **Descent method**

爲了求得nonlinear function的極小值，我們在這裡介紹descent method的作法。
Descent method的演算法如下：

Algorithm Descent method

begin
  $k := 0$; $\mathbf{x} := \mathbf{x}_0$; $found :=$ **false** {Starting point}
  while (**not** $found$) **and** ($k < k_{\max}$)
    $\mathbf{h}_d :=$ search_direction($\mathbf{x}$) {From $\mathbf{x}$ and downhill}
    if (no such $\mathbf{h}$ exists)
      $found :=$ **true** {$\mathbf{x}$ is stationary}
    else
      $\alpha :=$ step_length($\mathbf{x}, \mathbf{h}_d$) {from $\mathbf{x}$ in direction $\mathbf{h}_d$}
      $\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_d$; $k := k+1$ {next iterate}
end

這個演匴法從一個 initial guess x_0 開始，第一步先找一個 descent direction h_d，使得沿著這個方向走 F 的值會下降。再來，第二步就是要看沿著這個方向走多遠，會使得這個值下降的最多。反覆的執行上面的步驟，直到我們找不到一個方向會使得 F(x)的值下降爲止。我們希望最後我們找到的是一個局部極小值，但是這個演算法也有可能會停在其他的 stationary point，例如當 Hessian Matrix 爲 singular 時。

把 F(x)用泰勒展開式展開：

$$F(x+\alpha h) = F(x) + \alpha h^T F'(x) + O(\alpha^2)$$
$$\cong \mathrm{F(x)} + \alpha h^T F'(x) \quad \text{當}\alpha\text{夠小時}$$

如果在α 等於 0 的時候，$F(x+\alpha h)$ 是一個 decreasing function，則 h 就是一個 descent direction。

Descent direction 的定義如下：

Definition Descent direction.

$\mathbf{h}$ is a descent direction for F at $\mathbf{x}$ if $\mathbf{h}^\top \mathbf{F}'(\mathbf{x}) < 0$.

如果找不到這樣的 h，也就是$F'(x) = 0$ ，代表我們已經到了 stationary point。

- **Steepest descent method**

找 descent direction 的方法有好幾種，我們在這裡先介紹 steepest descent method。基本的想法是看看往哪個方向走，F(x)在單位距離下降的值最多。

$$\lim_{\alpha->0}\frac{F(x)-F(x+\alpha h)}{\alpha\|h\|}=-\frac{1}{\|h\|}h^T F'(x)=-\|F'(x)\|\cos\theta$$

爲 $F'(x)$ 和 $\frac{h}{\|h\|}$ 的夾角。

當 θ 等於 π 時,可以得到最小值。因此，steepest descent direction 爲 $h_{sd}=-F'(x)$ ，也就是 gradient $F'(x)$ 的反方向。

在一開始的時候，用這樣 steepest descent 的方法可以很快速的接近局部極小值，但是在快要到的時候，它會用 zig-zag 的方式移動，收斂的速度會變慢，很難到達真正的局部極小值。

.

- **Newton Method**

另一個尋找 descent direction 的方法是用 Newton Method。Newton Method 和 steepest descent method 不同的地方是在於決定 descent direction 的方法不同。Newton Method 是想要找到一個 h，使得 x+h 會到達局部極小值 。因爲它是局部極小值，所以 $F'(x+h)=F'(x^*)=0$ 。

依據泰勒展開式：

$$\begin{aligned}F'(x+h)&=F'(x)+F''(x)h+O(\|h\|^2)\\&\cong \mathrm{F'(x)}+F''(x)h \quad 當\|\mathrm{h}\|夠小時\end{aligned}$$

當 $F'(x+h_n)=0$ ，可得 $F''(x)h_n=-F'(x)$ ----(3)

只要求解這個線性系統，就能得到 h_n 的值。

Newton Method 在靠近局部極小值時，會收斂的比較快速，因爲在靠近局部極小值時，用二次式能夠得到對 F(x)比較好的近似，收斂速度比較快。但是在離局部極小值很遠時，這樣的近似比較不好，所以收斂速度比較慢。

- **Hybrid Method**

Steepest descent method 和 Newton Method 是互補的兩個方法。Steepest descent method 在離局部極小值比較遠時，收斂的速度比較快。Newton Method 在離局部極小值比較近時，收斂的速度比較快。

爲了提高收斂的速度，很自然的一個想法是用在離局部極小值比較遠時，用 steepest descent，在離局部極小值比較近時，用 Newton Method。

演算法如下：

$$
\begin{array}{l}
\textbf{if } \mathbf{F}''(\mathbf{x}) \text{ is positive definite} \\
\quad \mathbf{h} := \mathbf{h}_{\mathrm{n}} \\
\textbf{else} \\
\quad \mathbf{h} := \mathbf{h}_{\mathrm{sd}} \\
\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}
\end{array}
$$

在能用 Newton Method 時，也就是 $F''(x)$ 爲 positive definite 時，使用 Newton Method。在不能用 Newton Method 時，代表離局部極小值很遠，此時用 steepest descent method。

- **Line search**

在決定 descent direction h_d 後，再來就是要決定走多遠。因爲起始點 x 和要移動的方向 h 都已經決定，所以 $F(x+\alpha h)$ 現在變成 α 的函數。

$\varphi(\alpha) = F(x+\alpha h)$ x and h 爲定值, $\alpha \geq 0$

在決定 α 之後，就可以決定要走多遠。要求出 α 的值，可以解出 $\varphi(\alpha)$ 的最小值，但是這樣做速度可能會很慢。另一種可行的作法是只要找到一個 α 使得 $F(x+\alpha h) < F(x)$ 即可。可以利用 binary search 的方式來找到這樣子的 α 。

- **Levenberg-Marquardt**

之前的 descent method 是要解 nonlinear function 的 minimization 問題，Levenberg-Marquardt method 則是要解決 nonlinear function 的 least square 問題。基本上，Levenberg-Marquardt method 在離局部極值比較遠的時候，表現的像是 steepest descent method，在離局部極值比較近的時候，表現的像是 Newton Method。所以基本上 Levenberg-Marquardt method 就是一個 Hybrid Method，在 steepest descent method 和 Newton Method 中進行切換。

這個方法所找出來的 descent direction 如下：

$(F''(x)+\mu)h_{lm} = -F'(x)$ 　且 $\mu \ge 0$ 　----(4)

Levenberg-Marquardt method 就是利用控制這個 damping term μ 來決定是要切換到 Newton method 還是 steepest descent method。

μ 具有以下的性質：

(1) 因為 $\mu \ge 0$，所以 $(F''(x)+\mu)$ 一定是 positive definite 矩陣，這樣可以保證 h_{lm} 一定是 descent direction。

(2) 當 μ 夠大的時候，$h_{lm} \cong \dfrac{-F'(x)}{\mu}$，這就是一個 steepest descent direction。

(3) 當 μ 很小的時候，$h_{lm} \cong h_n$，這個方向就像是一個 Newton method 所找出來的方向。

所以只要控制這個 damping term μ，讓它在局部極值比較遠的時候，讓它的值比較大，讓它在離局部極值比較近的時候，讓它的值比較小，即可達到有如 Hybrid method 的效果。詳細的演算法如下：

```
Levenberg–Marquardt method
begin
  k := 0;   ν := 2;   x := x_0
  A := J(x)^T J(x);   g := J(x)^T f(x)
  found := (‖g‖_∞ ≤ ε_1);   μ := τ * max{a_ii}
  while (not found) and (k < k_max)
    k := k+1;   Solve (A + μI)h_lm = −g
    if ‖h_lm‖ ≤ ε_2(‖x‖ + ε_2)
      found := true
    else
      x_new := x + h_lm
      ϱ := (F(x) − F(x_new))/(L(0) − L(h_lm))
      if ϱ > 0                                  {step acceptable}
        x := x_new
        A := J(x)^T J(x);   g := J(x)^T f(x)
        found := (‖g‖_∞ ≤ ε_1)
        μ := μ * max{1/3, 1 − (2ϱ − 1)^3};   ν := 2
      else
        μ := μ * ν;   ν := 2 * ν
end
```

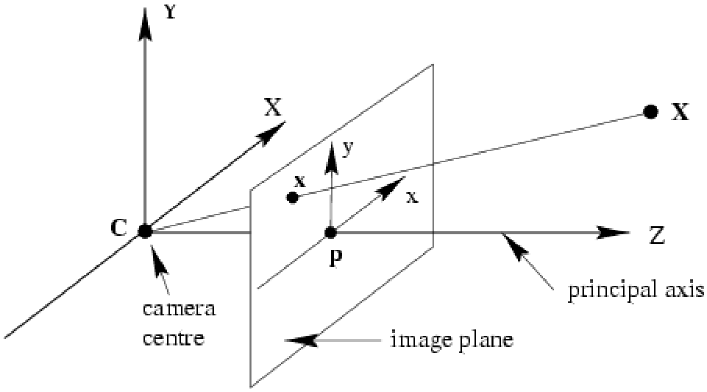
Camera projection models:

- **Pinhole camera**

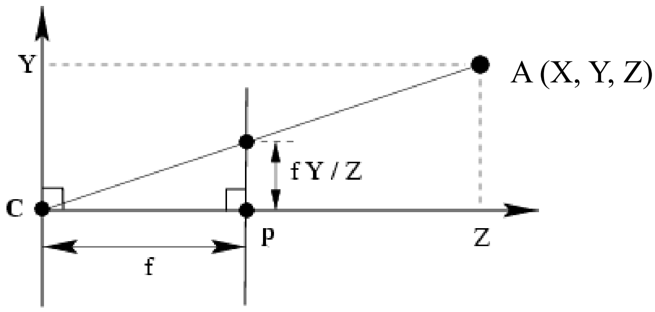
pinhole camera 是最簡單最理想的一種模型，光線僅僅只能從無限小的開孔進入，而形成一倒立影像在開孔面的對面平面上。

- **Pinhole camera model**

通常為了簡化，我們把成像平面放置在相機中心與觀測物體中間，如此會成一正立影像於成像平面上。



如圖，X 表示 3D 空間中的一個觀測點，會投影至成像平面上的 x 座標點。



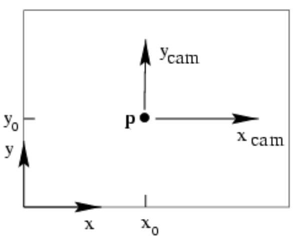
$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

將整個 model 投影至相機的 YZ 平面上，利用相似三角形，可以算出 3D 空間中點 A 的 Y 座標會投影至成像平面 fY/Z 位置，同理 X 座標會投影至 fX/Z 位置。因此可以以下面的投影矩陣表示此投影關係。

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$,等於 $$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- **Principal point offset**



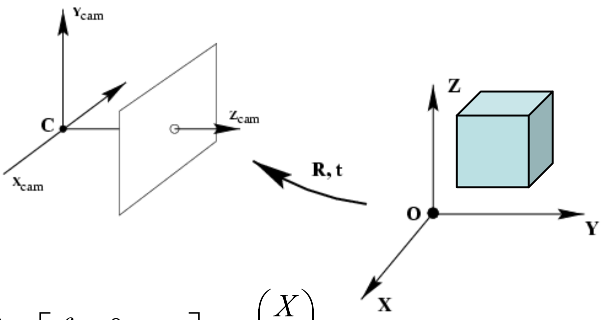
如將成像平面座標原點從相機中心(x0,y0)移至左下角，投影矩陣改變為：

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{K}=\begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$
我們稱此 K 為理想上的相機內在參數矩陣(Intrisic matrix)

$$\mathbf{K}=\begin{bmatrix} fa & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$
若考慮 non-square pixels、skew、radial distortion 問題，我們會在 K 上再加入 aspect ratio (a), skew (s)參數

- **Camera rotation and translation**



$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R}_{3\times 3}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

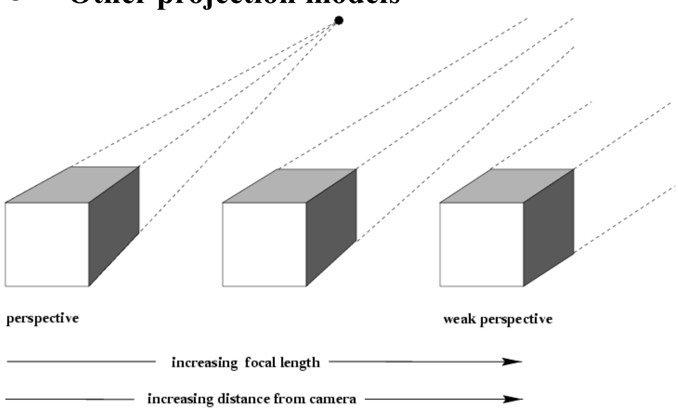
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{R}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$
我們稱[R|t]為相機外在參數矩陣

- **Two kinds of parameters**

因此相機參數主要可以分成兩類：

1. 內在參數(internal/intrinsic parameters)：包含 focal length、optical center、aspect ratio，代表著相機的種類。
2. 外在參數(external/extrinsic parameters)：包含 rotation、translation，代表著相機擺放的位置。

- **Other projection models**

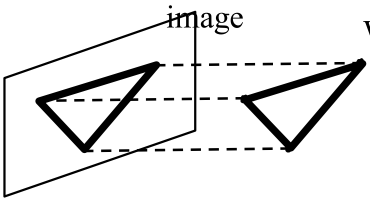


Weak perspective：(x,y,z)➔s(x,y)

對任意點來說，s 係數皆相同，另外平行線將不會收斂至一點上，而會保持平行，對於越小且越遠的物體，會越逼近於此種投影 model。

- **Orthographic projection**

當投影中心與投影平面之間距離為無限大時的投影 model，也稱為 parallel projection



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

- **Other types of projection**

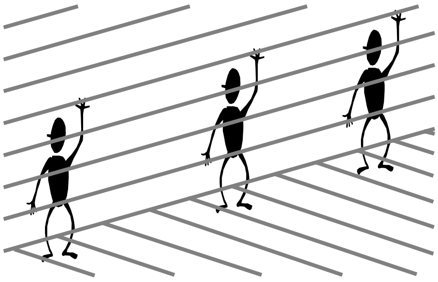
Scaled orthographic (weak perspective)：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

Affine projection (paraperspective)：

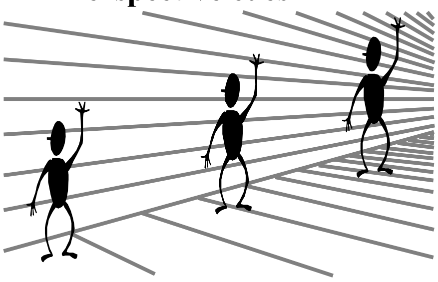
$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

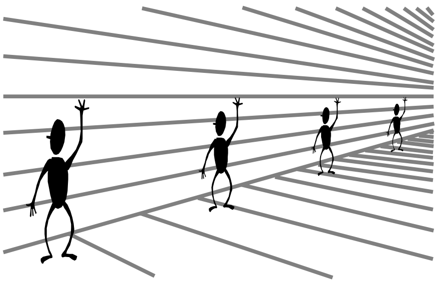
- **Fun with perspective**



若是將線條平行繪製，會讓人在視覺上感覺三個人物是一樣的大小。

- **Perspective cues**


(a)

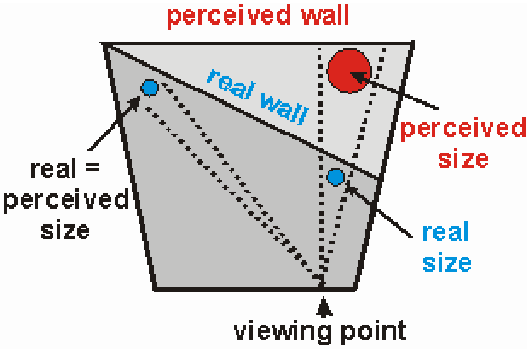

(b)

若將線條繪製得有空間感，

(a) 影像上相同大小的三個人會讓人在視覺上感覺不同。

(b) 影像上不同大小的四個人會讓人在視覺上感覺相同。

- **Fun with perspective**



另一種利用 perspective 概念欺騙人類視覺的效果

- **Forced perspective in LOTR**

在魔戒拍攝中，許多與哈比人的互動片便利用這種概念處理

<u>Camera calibration:</u>

- **Camera calibration**

在 camera 裡面有兩種資訊是我們希望去知道的

1. camera 在那裡 ？
2. camera projection model 裡面那些參數是什麼 ？(focal length, optical center, aspect ratio)

Camera calibration 主要的方法有兩大類：

1. Photometric calibration: use reference objects with known geometry.
2. Self-calibration: only assume static scene, e.g. structure from motion.

Self-Calibration:

假設拍攝的場景是靜態的，只用靜態的場景來估計 camera 的參數

估計的方法：bundle adjustment

Photometric Calibration:

在影片中放置參考物件，並且確實知道這個參考物件在 3D 中的位置，大小等 geometry information。Photometric calibration 比較簡單，但通常參考物件是你不希望出現在影片中的東西，所以必須再做額外的處理 (e.g. In-panting)

已知：

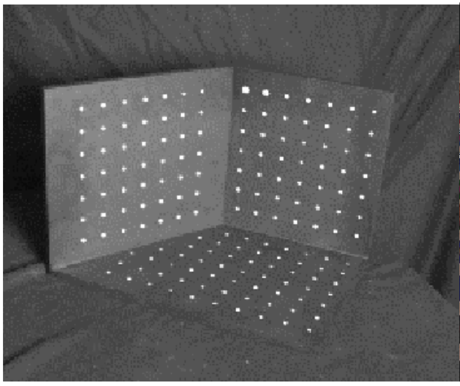
a. 參考物件在 3D 中的位置，大小等 geometry information
b. 在 image 中特定點的 2D 座標及其對應的 3D 座標點，

目標：

嘗試去估計 camera projection matrix

在 Photometric calibration 類型中主要有三種方法：

a. Linear regression (least squares)
b. Nonlinear optimization
c. Multiple planar patterns




Chromaglyphs (HP research)

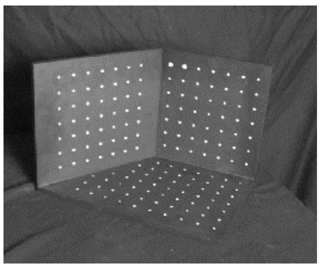
HP Lab 用來校正相機 pose 的一種 Calibration pattern

- **Linear Regression (least squares)**

1. camera 共有 11 個 parameters 需要去估計。
2. 每個點的 2d/3d 對應關係給了我們兩組 equation。
3. 所以我們至少需要 6 個點去 recover 這 11 個參數，當然點多一些會有幫助，因爲我們還需考量雜訊及誤差的問題，更多的點可以幫助我們更精確的估計。

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



利用已知的 3D 參考座標點(Xi,Yi,Zi)，和影像中所測量到對應的 feature 位置(ui,vi)，來估測 M 矩陣裡的 11 個未知數(m00~m22)。

$$
\begin{aligned}
u_i &= \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1} \\
v_i &= \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}
\end{aligned}
$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

之後利用 least-square 的方式來求解 M。

主要壞處：分不出內在與外在參數。

- **Normal equation**

給定一個 overdetermined system：Ax = b

(註：A 矩陣的高度爲 n，寬度爲 k，並且 n>k，稱之爲 overdetermined system)

Normal equation 爲求解 x 的近似值，以滿足：$\mathbf{A^tAx = A^tb}$

優點：

1. 所有未知參數集中在解一矩陣 M
2. 可以估測出 3D 空間中的點會投影到影像上哪個位置

缺點：

1. 無法求出相機中某一指定參數
2. 無法分隔出內在與外在參數

- **Nonlinear Optimization**

基本上我們可以用 Levenberg-Marquardt (Nonlinear least square method) 來求解

$$
\begin{aligned}
u_i &= f(\mathrm{M}, \mathbf{x}_i) + n_i = \hat{u}_i + n_i, \quad n_i \sim N(0, \sigma) \\
v_i &= g(\mathrm{M}, \mathbf{x}_i) + m_i = \hat{v}_i + m_i, \; m_i \sim N(0, \sigma)
\end{aligned}
$$

xi 是三度空間中的座標值，ui, vi 是螢幕上的座標點

M 是我們要求的，我們要找一個 M 使得 ui,vi 出現的機率最大

基本上可以把它想成一個機率的模式，在什麼情況下 ui,vi 出現的機率最大，而這邊我們用一個 Gaussian model 來考慮它機率的分布

$$
\begin{aligned}
L &= \prod_i p(u_i|\hat{u}_i)p(v_i|\hat{v}_i) \\
&= \prod_i e^{-(u_i-\hat{u}_i)^2/\sigma^2} e^{-(v_i-\hat{v}_i)^2/\sigma^2}
\end{aligned}
$$

Log likelihood of $\boldsymbol{M}$ given $\{(u_i, v_i)\}$

$$C = -\log L = \sum_i (u_i - \hat{u}_i)^2/\sigma_i^2 + (v_i - \hat{v}_i)^2/\sigma_i^2$$

我們要 minimize C。我們要找一個 M 來 optimize 這個 C，問題在於說，這個 ui 它本身是一個 M 的 nonlinear function。所以我們要用 nonlinear least square 的 method，這邊我們可用 Levenberg-Marquardt method

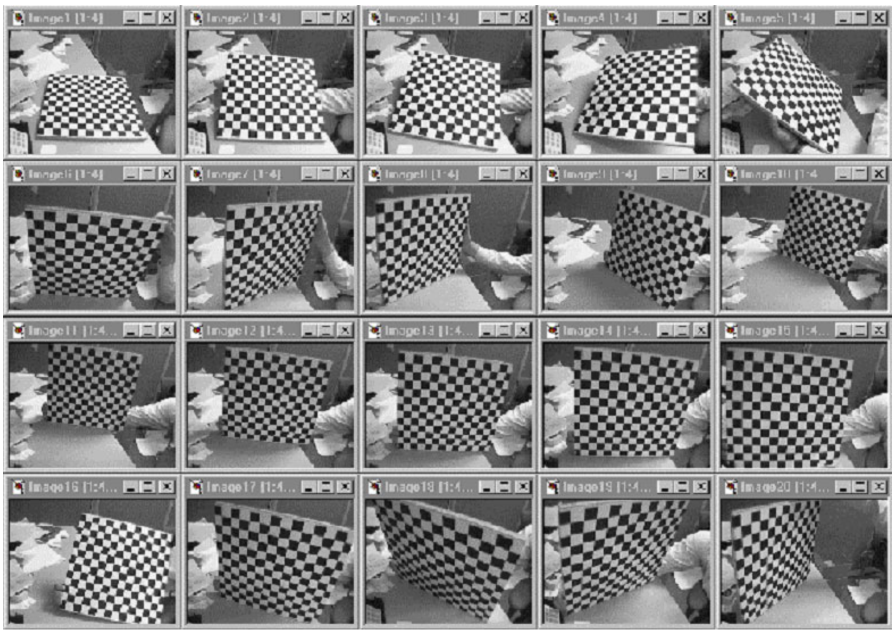
- **Multi-Plane Calibration**

之前的方法(Linear regression, Nonlinear optimization)都要明確知道參考物件空間中的座標點，multi-plane calibration 只要假設參考物件是一個平面物件，不需要知道參考物件空間中的座標點。

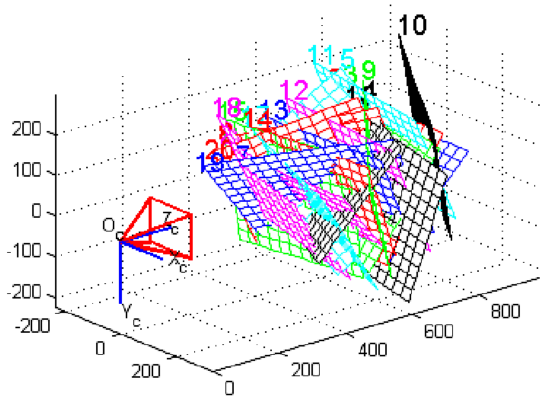
優點：

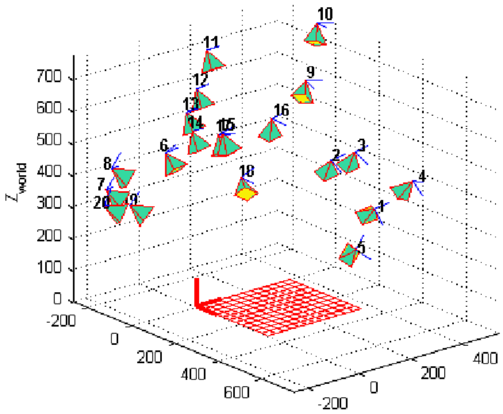
1. 只需要一個參考平面
2. 不用去知道空間中的位置和方向
3. 已經有現成的 source code 可以使用
   a. Intel's OpenCV library: http://www.intel.com/research/mrl/research/opencv/
   b. Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
   c. Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/

首先要準備一個 check board pattern，然後 camera 不動，轉動參考平面，得到如下的影像，就可以求出 camera intrinsic and extrinsic parameters



或是反過來，參考平面不動，變換 cmaera 的位置角度，這樣也可以得到類似的影像，也可以求出 camera intrinsic and extrinsic parameters





這個方法唯一要手動的部份是在第一張 image 的時候要手動點出四個 corner points，其他都是自動就會做好。

應用：
3D scanning image shadow
測量某個物件的 3D geometry，但通常這需要一個比較複雜的系統，Bouget 在他的博士論文中做出一個系統，可以讓我們大略的估計這個物件的 3D geometry。他用的方法說明如下：

放一台 camera 及一個抬燈，然後用一個直的棍子讓它的影子投影在物體上，看這個棍子的影子在這個物體上怎樣做扭曲，這樣就能夠得到這個物件的 geometry，用的方法就是所謂的 triangulation。

1. 首先要求得光源的方向和 camera projection matrix。
2. 棍子影子在影像中的每個點，你都明確知道在空間中是屬於那條線，而光是走直線 (directional light)，因此這兩條線的交點就是這個點在空間中的位置

- **Bundle adjustment**

1. Bundle adjustment 是一個可以同時估計 3D geometry 和 camera parameters 的技巧，它可以使用不同的 projection matrix, 不同的 noise model。
2. Bundle adjustment 是一個做 Self-Calibration 的技巧，它假設場景是靜態的，利用場景中的 feature points 來想辦法估計取得 camera 的 intrinsic 或 extrinsic matrix parameters.
3. 在做 panoramas 時，同時考慮所有 image, 跟所有 corresponded pair，把這個 drift 做一個 global optimization。
4. 對 match-move , panoramas, bundle adjustmen 都是一個最終的最佳化流程。

輸入：

1. n 個 3d 座標點，不需要知道它們的座標值，但是必須知道被投影到螢幕上的那一點，這用 feature match 就可達到。
2. m 個 view，必須知道每個點在每個 view 中的 2d 座標值是什麼。

- n 3D points are seen in m views
- x_{ij} is the projection of the i-th point on image j
- a_j is the parameters for the j-th camera
- b_i is the parameters for the i-th point
- **BA attempts to minimize the projection**

$$\min_{\mathbf{a}_j,\mathbf{b}_i}\sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_j,\mathbf{b}_i),\ \mathbf{x}_{ij})^2$$

predicted projection

Euclidean distance

error

Bundle adjustment 就是要嘗試去找出 aj (parameters for the j-th camera) 和 bi (parameters for the i-th point) 使得 predicated projection 和實際影像的差距爲最小，而 predicated projection 就是 camera projection matrix。所以也就是解一個 nonlinear least square，所應用的方法也是 Levenberg-Marquardt method。